From Sets of Good Redescriptions to Good Sets of Redescriptions

Janis Kalofolias, Esther Galbrun, Pauli Miettinen

From Sets of Good Redescriptions to Good Sets of Redescriptions

Janis Kalofolias
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
kalofolias@mpi-inf.mpg.de

Esther Galbrun
Inria Nancy – Grand Est
Nancy, France
esther.galbrun@inria.fr

Pauli Miettinen
Max Planck Institute for Informatics
Saarland Informatics Campus, Germany
pauli.miettinen@mpi-inf.mpg.de

***Abstract*—Redescription mining aims at finding pairs of queries over data variables that describe roughly the same set of observations. These redescriptions can be used to obtain different views on the same set of entities. So far, redescription mining methods have aimed at listing all redescriptions supported by the data. Such an approach can result in many redundant redescriptions and hinder the user's ability to understand the overall characteristics of the data.**

In this work, we present an approach to find a good set of redescriptions, instead of finding a set of good redescriptions. That is, we present a way to remove the redundant redescriptions from a given set of redescriptions. We measure the redundancy using a framework inspired by the subjective interestingness based on maximum-entropy distributions as proposed by De Bie in 2011. Redescriptions, however, raise their unique requirements on the framework, and our solution differs significantly from the existing ones. Notably, our approach can handle disjunctions and conjunctions in the queries, whereas the existing approaches are limited only to conjunctive queries. The framework also reduces the redundancy in the redescription mining results, as we show in our empirical evaluation.

I. Introduction

Redescription mining is a data mining task that aims at finding alternative characterizations of (roughly) the same objects. The motivation behind this is simple and intuitive: If some objects can be described in alternative ways, then they form a particularly coherent group. Furthermore, identifying such alternative descriptions tells us something about the properties appearing in such synonymic characterizations.

Take for instance the case of areas of the globe, for which climatic information as well as records of observed animal species are available. Areas that share a particular climatic profile and host particular species form a coherent group, closely related to the concept of an environmental niche in ecology. As another example, in medicine, identifying groups of patients who share similar profiles might help relate genetic traits, disease symptoms, and treatment outcomes. The same idea can also be of interest in other fields, including ethnography, sociology, or chemistry, for instance.

Like many other data mining tasks, redescription mining is subject to the issue of pattern explosion, whereby a large number of results are returned by the mining algorithm – many of them slight variations of one another – and we face the challenge of identifying an interesting non-redundant subset for further inspection.

Thus, the problem we address in this paper is one of pattern selection. Considering a dataset that consists of a pair of matrices over the same objects, and given a collection of redescriptions that has been mined from this dataset, we want to select a redescription set that is informative and non-redundant. Note that we are not introducing any new pattern formalism nor any new algorithm for mining redescriptions. Instead, we focus on the problem of ranking and filtering redescriptions as a post-process.

One approach to evaluating the interestingness of patterns, which we adopt here, is to use them to construct a statistical model of the data. A pattern can be seen as an observation from the data and can be evaluated against the current model. If the current model already accounts for the observation, i.e. the pattern does not provide any new information about the data, it is considered to be redundant and can be discarded. Otherwise, the pattern is deemed interesting and is integrated into the model, increasing its quality.

Our approach is iterative: at each step the candidate redescriptions are evaluated against the model that encapsulates the current knowledge available about the data, i.e. the information about the dataset acquired through the redescriptions selected so far. We can compute a score for each candidate indicating how much novel information it contains with respect to the current state, allowing us to rank the redescriptions. We then select the most informative one and incorporate it into the model. Next, we proceed with updating the scores of the candidates and selecting the best one, until no additional information can be incorporated into the model; that is, until only redundant candidate redescriptions are left.

Our modeling techniques rely on the Maximum Entropy distribution and the framework of subjective interestingness, as proposed by De Bie [6]. In brief, we keep up a maximum entropy distribution conditioned on all of the already-seen redescriptions to model what is already known, and consider redundant all redescriptions that have a high likelihood under this distribution. We emphasize, however, that our model differs from that of [6] and subsequent work (e.g. [21]) in significant ways, as the patterns we are studying – namely redescriptions – are more intricate than the patterns studied in prior work.

The approach presented here rests on two main lines of work. *Redescription mining*, on one hand, provides us with a rich pattern language, while *maximum entropy modeling*, on the

other hand, provides us with well-grounded selection principles. The next section where we cover the related research, is divided between these two domains upon which we build.

II. Related Work

A. Redescription Mining

Since the introduction of redescription mining by Ramakrishnan et al. in 2004 [31], a few problem variants [8], [28], [36], several algorithms [9], [13], [31], [37], as well as an interactive mining tool [11], [12] have been introduced. Redescription mining is a descriptive approach, tailored towards exploratory data analysis. Redescriptions are local patterns in the sense that they characterize subsets of the data, rather than capturing properties of the dataset as a whole. Redescription mining is related to rule mining techniques that aim at discovering association rules [2] or subgroups [27], for instance, and to classification and clustering techniques including subspace clustering [1], [22] and multi-view clustering [5], among others. However, its goal of finding descriptive, interpretable patterns across several datasets is a distinguishing feature.

B. Maximum Entropy modeling

The maximum entropy principle, formalized by Jaynes [7], allows to build a probability distribution over possible datasets under constraints, with the advantage of not adding extra bias beyond what is assumed from the constraints.

It is a versatile principle which has been applied to ecological modeling [30] as well as to natural language processing (NLP) [4], and has been used to construct a condensed representation of the dataset to be used for query approximation [25], [29] or to compute degrees of belief [14].

De Bie [6] proposed to use maximum entropy models to evaluate *subjective interestingness* and in the present work we follow his general idea. Importantly, unlike the previous work (e.g. [6], [21], [32]), we need to also handle disjunctions, rather than just conjunctions of the variables. On the other hand, similarly to most existing approaches [16], [32], we apply ranking and filtering as a post-processing, meaning that we require as an input a collection of candidate patterns. Mampaey et al. [24] proposed a method for finding interesting itemsets which mines candidates on the fly.

A further distinction is between approaches which model rows of the dataset [24], [32] versus those which consider the dataset as a whole [6], handling increasingly complex priors and pattern types [18]–[20]. Related modeling approaches include the use of Markov Random Fields [35] and the minimum description length (MDL) principle [33], [34], but do not offer a similar flexibility and modeling power.

III. Background

A. Maximum Entropy modeling

The maximum entropy distribution p subject to the constraints $C = \{(f_c, \pi_c)\}$ is defined as the solution to the following program:

$$\max_{p \in \mathcal{P}} - \int_{\mathcal{V}} p(\boldsymbol{x}) \log p(\boldsymbol{x}) d\boldsymbol{x} \tag{1}$$

$$\text{such that} \quad \int_{\mathcal{V}} p(\boldsymbol{x}) f_c(\boldsymbol{x}) d\boldsymbol{x} = \pi_c \qquad \forall c \in C. \tag{2}$$

The optimisation is over the set $\mathcal{P}$ of distributions p over $\mathcal{V}$ (i.e. such that $p(\boldsymbol{x}) \geq 0$ and $\int_{\mathcal{V}} p(\boldsymbol{x}) d\boldsymbol{x} = 1$).

The solution to this optimisation problem can be computed using standard tools from constrained variational optimisation in order to obtain

$$p_\lambda(\boldsymbol{r}) = \frac{1}{Z_{\lambda_c}} \exp\left(\sum_{c \in C} \lambda_c f_c(\boldsymbol{r})\right), \tag{3}$$

where λ_c are the Lagrange multipliers for the constraints and Z_{λ_c} is the partition function

$$Z_{\lambda_c} = \int_{\mathcal{V}} \exp\left(\sum_{c \in C} \lambda_c f_c(\boldsymbol{x})\right) d\boldsymbol{x}\ . \tag{4}$$

The Lagrange multipliers $\boldsymbol{\lambda}$ in Eq. (4) can be computed as the solution of the convex dual

$$q(\boldsymbol{\lambda}) = \log Z_{\boldsymbol{\lambda}} - \sum_{c \in C} \lambda_c \pi_c,$$

which can be shown to be equivalent to the conditions

$$\mathbb{E}_p\left[f_c(\boldsymbol{r})\right] = \pi_c \qquad \forall c \in C. \tag{5}$$

B. Redescription Mining

In the formulation of redescription mining considered here, the input data consist of entities with two sets of characterizing variables, thus forming a dataset with two sides. We generally refer to these two sides as the left and right-hand sides, and represent them using two matrices $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$ over two sets of variables $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$, respectively. The domains of the variables are denoted by $\mathcal{V}_{\mathbf{L}}$ and $\mathcal{V}_{\mathbf{R}}$, respectively, and they can be either continuous or discrete. The set of entities characterized by the two sides is denoted by E, thus both matrices have $|E|$ rows. The value of $D(i, j)$ is the value of variable $v_j \in V$ for entity $d_i \in E$.

The output of redescription mining consists of a collection of query pairs, the redescriptions, of the form $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$, where the queries $q_{\mathbf{L}}$ and $q_{\mathbf{R}}$ are *logical statements* over the variables in $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$, respectively.

Truth assignments are then defined over the variables by requiring them to take values within a subset of their domain. We denote these truth assignments using Iverson notation, e.g. $[a \leq v \leq b]$, except for Boolean variables where the truth assignment $[v = \texttt{True}]$ is denoted simply as v. These truth assignments and their negations are combined into logical statements using the Boolean operators $\wedge$ (and) and $\vee$ (or).

The *support* of a query q is the subset of entities for which the query holds true, that is $\mathrm{supp}(q) = \{d_i : q \text{ is true for } d_i \in D\}$. The support of a redescription R is the set of entities that satisfy both logical statements $\mathrm{supp}(R) = \mathrm{supp}(q_{\mathbf{L}}) \cap \mathrm{supp}(q_{\mathbf{R}})$. The set of variables over which q is expressed is denoted as $\mathrm{vars}(q)$.

To measure the *accuracy* of a redescription R, we use the Jaccard coefficient of the supports of the queries

$$\mathrm{J}(R) = \mathrm{J}(q_{\mathbf{L}}, q_{\mathbf{R}}) = \frac{|\mathrm{supp}(q_{\mathbf{L}}) \cap \mathrm{supp}(q_{\mathbf{R}})|}{|\mathrm{supp}(q_{\mathbf{L}}) \cup \mathrm{supp}(q_{\mathbf{R}})|} ,$$

being at once a simple, symmetric, and intuitive measure.

Accuracy, however, is a *local* feature of the redescription and does not help in finding a good *set* of redescriptions. Therefore, we need to look at the variables and values at play in the logical statements of the redescriptions. We will use these logical statements to constrain a maximum entropy distribution over possible datasets, allowing us to evaluate the probability of observing a certain redescription given those we already know to be present.

Most interesting in a redescription is the association between the two queries, and the entities that satisfy both of them. For this reason, our main objective here is in modeling the conjunction of the two queries that make up a redescription, and the intersection of their supports that is simply the support of the redescription. Henceforth, we will treat a redescription $R = (q_{\mathbf{L}}, q_{\mathbf{R}})$ as the logical statement $s = q_{\mathbf{L}} \wedge q_{\mathbf{R}}$, with its associated support $\mathrm{supp}(s) = \mathrm{supp}(q_{\mathbf{L}}) \cap \mathrm{supp}(q_{\mathbf{R}}) = \mathrm{supp}(R)$, evaluated over the dataset D resulting from the concatenation of $D_{\mathbf{L}}$ and $D_{\mathbf{R}}$; similarly, the variables in $V_{\mathbf{L}}$ and $V_{\mathbf{R}}$ are pooled together to form V. Thus, our dataset D is a matrix with $N = |V|$ columns and $M = |E|$ rows.

IV. Theory

We now introduce our probabilistic models for the data. First, we explain how to model the values occurring within an arbitrary row while accounting for the presence of a given set of statements, using the maximum entropy principle. Next, we explain how to represent an entire dataset based on this row model, using a mixture model that comes in two variants.

A. Modeling Rows

Let $\boldsymbol{r}$ represent a vector sampled from the domain of the variables $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_N$. Our goal in this section is to define $p(\boldsymbol{r} \mid S)$, the probability that an arbitrary dataset row takes on values $\boldsymbol{r}$, assuming the presence of a given collection of logical statements S. We denote this probability simply as $p(\boldsymbol{r})$ when the constraining set of statements is clear from the context. Each statement $s \in S$ is associated to a probability π_s that a random instance $\boldsymbol{r}$ satisfies it, with respect to the probability measure defined by the model $p_{\mathcal{M}}$. Thus, in our setting, the conditions of Eq. (5) can be specified formally as

$$\mathbb{E}_{p_{\mathcal{M}}}\left[\chi_s(\boldsymbol{r})\right] = \pi_s \quad \forall s \in S , \tag{6}$$

where $\chi_s(\boldsymbol{r})$ is the characteristic function of the statement s: $\chi_s(\boldsymbol{r}) = 1$ if s holds true on $\boldsymbol{r}$, and $\chi_s(\boldsymbol{r}) = 0$ otherwise.

We can assume that these conditions are satisfiable since they are derived from actual observations. Then, plugging them as constraint functions into Eq. (3) and (4), we obtain the solution and partition function

$$p_{\boldsymbol{\lambda}}(\boldsymbol{r}) = \frac{1}{Z_{\boldsymbol{\lambda}}} \exp\left(\sum_{s \in S} \lambda_s \chi_s(\boldsymbol{r})\right) , \tag{7}$$

$$Z_{\boldsymbol{\lambda}} = \int_{\mathcal{V}} \exp\left(\sum_{s \in S} \lambda_s \chi_s(\boldsymbol{x})\right) d\boldsymbol{x} . \tag{8}$$

However, this formal solution to the maximum entropy optimisation problem cannot be used directly. Indeed, in this general form the model is computationally too complex and we need to exploit the problem structure to simplify it.

Factorising the distribution: In its general form, the integration in the partition function of Eq. (8) runs over the domain of the entire set of variables, whereas each logical statement typically only involves a small subset of them. Therefore, to simplify the computation we can split the statements into groups, so that the computation of each group involves as few variables as possible and can be carried out independently. The requirement for this is that the sets of variables appearing in the different groups be disjoint from one another, which also allows the sums within the exponents of Eq. (7) and (8) to be similarly split into groups.

Hence, we define a partitioning $\mathcal{K}$ of the variables V such that for all $s \in S$ there exists a $K \in \mathcal{K}$ with $\mathrm{vars}(s) \subset K$, and denote by $S_K = \{s \in S : \mathrm{vars}(s) \subset K\}$ the subset of statements that only contain variables in K. Then, the sets $\{S_K, K \in \mathcal{K}\}$ form a partitioning of S and we can write

$$\exp\left(\sum_{s \in S} \lambda_s \chi_s(\boldsymbol{r})\right) = \prod_{K \in \mathcal{K}} \exp\left(\sum_{s \in S_K} \lambda_s \chi_s(\boldsymbol{r})\right) .$$

The integral in the partition function can also be split into integrals over domains $\mathcal{V}_K$ of the variables in K. We obtain

$$p_{\boldsymbol{\lambda}}(\boldsymbol{r}) = \prod_{K \in \mathcal{K}} p_{\boldsymbol{\lambda}_K}(\boldsymbol{r}) , \tag{9}$$

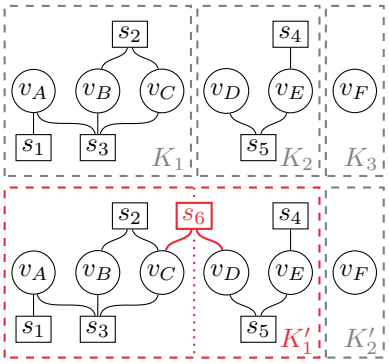
which is a product of sub-probabilities defined over independent subsets of the original variables

$$p_{\boldsymbol{\lambda}_K}(\boldsymbol{r}) = \frac{1}{Z_K(\boldsymbol{\lambda})} \exp\left(\sum_{s \in S_K} \lambda_s \chi_s(\boldsymbol{r})\right)$$

$$Z_K(\boldsymbol{\lambda}) = \int_{\mathcal{V}_K} \exp\left(\sum_{s \in S_K} \lambda_s \chi_s(\boldsymbol{x})\right) d\boldsymbol{x} ,$$

and the normalisation can be performed per each $Z_K(\boldsymbol{\lambda})$ term.

Example. Consider a dataset over six variables, $\{v_A, \ldots, v_F\}$, and a collection of five logical statements over these variables $S = \{s_1, \ldots, s_5\}$, as shown in Fig. 1 (right). In this case, the finest partitioning of the variables, so that all statements are completely contained within one cluster, consists of the 3 groups $K_1 = \{v_A, v_B, v_C\}$, $K_2 = \{v_D, v_E\}$, and $K_3 = \{v_F\}$, as shown in Fig. 1 (top). Suppose that at this point the constraint $s_6 = v_C \vee \neg[v_D < 30]$ is added. To fulfill the containment requirement, we now need to merge K_1 and K_2, resulting in the clustering shown in Fig. 1 (bottom).



$$v_A, v_B, v_C \in \{0,1\}$$
$$v_D \in [10,50]$$
$$v_E \in [0,5]$$
$$v_F \in \{0,1\}$$

$$s_1 = v_A$$
$$s_2 = v_B \wedge v_C$$
$$s_3 = v_A \vee v_B \vee v_C$$
$$s_4 = [2 < v_E]$$
$$s_5 = [20 < v_D] \vee [2 < v_E]$$
$$s_6 = v_C \vee \neg [v_D < 30]$$

Figure 1: Clustering of the six data variables $v_A, \ldots, v_F$ of a row model with five logical statements $s_1, \ldots, s_5$ (top) and after the addition of a sixth statement s_6 (bottom).

To further speed-up the computation, we can also perform a re-parametrization (or factorisation) of the distribution, for instance using the Junction Tree algorithm [17].

Quantising the domains: A further source of complexity results from integrating over the domain of the variable set. We can exploit the fact that the redescriptions consider intervals over the continuous variables to quantise the variables into discrete bins defined over these intervals.

Looking at the truth table for the statements in S, we group together the regions of the domain where the same combination of statements are satisfied. Formally, adopting an arbitrary ordering $s_1, \ldots, s_\sigma$ of the statements in S, for a vector $(t_1, \ldots, t_\sigma) \in \{0,1\}^\sigma$ of truth values for the statements we denote the corresponding collection of regions as

$$T_{t_1 \cdots t_\sigma} = \{\boldsymbol{r} \in \mathcal{V} : \chi_{s_i}(\boldsymbol{r}) = t_i \text{ for all } s_i \in S\} \ .$$

The resulting partitioning of the domain is denoted as $\mathcal{T}$. Using a similar notation for the subscripts, we denote the combinations of Lagrange multipliers as $\zeta_{t_1 \cdots t_\sigma}(\boldsymbol{\lambda}) = \lambda_1 \cdot t_1 + \cdots + \lambda_\sigma \cdot t_\sigma$.

We can now express Eq. (7) and (8) as

$$p_{\boldsymbol{\lambda}}(\boldsymbol{r}) = \frac{1}{Z(\boldsymbol{\lambda})} \exp\left(\zeta_{\boldsymbol{t}(\boldsymbol{r})}(\boldsymbol{\lambda})\right) , \tag{10}$$

$$Z(\boldsymbol{\lambda}) = \sum_{T_{\boldsymbol{t}} \in \mathcal{T}} \int_{T_{\boldsymbol{t}}} \exp\left(\zeta_{\boldsymbol{t}}(\boldsymbol{\lambda})\right) d\boldsymbol{x} \tag{11}$$
$$= \sum_{T_{\boldsymbol{t}} \in \mathcal{T}} |T_{\boldsymbol{t}}(\boldsymbol{\lambda})| \exp\left(\zeta_{\boldsymbol{t}}(\boldsymbol{\lambda})\right) ,$$

where $\boldsymbol{t}(\boldsymbol{r}) = (\chi_{s_1}(\boldsymbol{r}), \ldots, \chi_{s_\sigma}(\boldsymbol{r}))$ is the truth vector of the statements for the current point.

Example. Going back to our example, consider the computation of the factor associated to group $K_1 = \{v_A, v_B, v_C\}$, involving statements s_1, s_2 and s_3 (see Fig. 1). The truth table of these statements is shown in Tab. I (left). The terms can be gathered based on the satisfiability of the constraints in the just 5 different combinations of Tab. I (right). The partition function of Eq. (11) can then be written with one term per combination, as

$$Z_{K_1}(\boldsymbol{\lambda}) = 1 \cdot e^{\zeta_{000}} + 2 \cdot e^{\zeta_{001}} + 1 \cdot e^{\zeta_{011}} + 3 \cdot e^{\zeta_{101}} + 1 \cdot e^{\zeta_{111}} .$$

Table I: Domain quantisation for the statements $\{s_1, s_2, s_3\}$ over the Boolean variables $\{v_A, v_B, v_C\}$ from K_1. Statements truth table (left) and accumulated terms (right).

v_A	v_B	v_C	χ_{s_1}	χ_{s_2}	χ_{s_3}	$T_{\boldsymbol{t}}$
0	0	0	0	0	0	T_{000}
1	0	0	1	0	1	T_{101}
0	1	0	0	0	1	T_{001}
1	1	0	1	0	1	T_{101}
0	0	1	0	0	1	T_{001}
1	0	1	1	0	1	T_{101}
0	1	1	0	1	1	T_{011}
1	1	1	1	1	1	T_{111}

$T_{\boldsymbol{t}}$	$\lvert T_{\boldsymbol{t}} \rvert$	$\zeta_{\boldsymbol{t}}$
T_{000}	1	0
T_{001}	2	λ_3
T_{011}	1	$\lambda_2 + \lambda_3$
T_{101}	3	$\lambda_1 + \lambda_3$
T_{111}	1	$\lambda_1 + \lambda_2 + \lambda_3$

Table II: Domain quantisation for the statements $\{s_4, s_5\}$ over the real-valued variables $\{v_D, v_E\}$ from K_2. From statement truths to region probabilities $P(Q) := \int_Q p(\boldsymbol{x}) d\boldsymbol{x}$.

Q	v_D	v_E	χ_{s_4}	χ_{s_5}	$T_{\boldsymbol{t}}$	$\lvert T_{\boldsymbol{t}} \rvert$	$\zeta_{\boldsymbol{t}}$	$p(\boldsymbol{x})$	$P(Q)$
Q_1	$[10,20]$	$[0,2]$	0	0	T_{00}	$10 \cdot 2$	0	$1/40$	0.5
Q_2	$[20,50]$	$[0,2]$	0	1	T_{01}	$30 \cdot 2$	λ_5	$1/(40 \cdot 15)$	0.1
Q_3	$[10,20]$	$[2,5]$	1	1	T_{11}	$10 \cdot 3$	$\lambda_4 + \lambda_5$	$2/(40 \cdot 15)$	0.1
Q_4	$[20,50]$	$[2,5]$	1	1		$+30 \cdot 3$			0.3

In this small example, the number of terms has been reduced from $2^3 = 8$ to 5. As the number of variables per factor rises, this simplification can become more dramatic.

In the more general case involving non-Boolean variables, the boundaries delimiting the regions of the domain can be easily identified from the thresholds of the intervals.

Example. Consider now the computation of the factor associated to group $K_2 = \{v_D, v_E\}$, involving the statements s_4 and s_5 of Fig. 1, respectively; we assume satisfiability probabilities $\pi_{s_4} = 0.4$ and $\pi_{s_5} = 0.5$, and use domains $v_D \in [10,50]$ and $v_E \in [0,5]$.

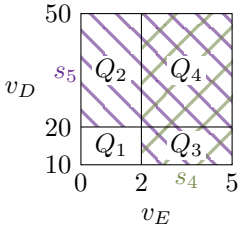


Figure 2: Regions of the domains of v_D and v_E.

The literals involved are $[2 < v_E]$ and $[20 < v_D]$, so the relevant thresholds are $(0, 2, 5)$ for v_E and $(10, 20, 50)$ for v_D, respectively. The resulting quantisation is depicted in Fig. 2, where each of the four blocks represents one region of the domain, denoted as $Q_1, \ldots, Q_4$. For each region, the satisfiabilities of both statements are shown in Tab. II. This combination determines the partition $T_{\boldsymbol{t}}$ to which the region is assigned. The measure $|T_{\boldsymbol{t}}|$ of a partition is simply the sum of the areas of the regions it consists of. Using these partitions and the corresponding combinations of Lagrange multipliers, we can write the partition function as

$$\begin{aligned} Z_{K_2}(\boldsymbol{\lambda}) &= \int_{T_{00}} e^{\zeta_{00}(\boldsymbol{\lambda})} d\boldsymbol{x} + \int_{T_{01}} e^{\zeta_{01}(\boldsymbol{\lambda})} d\boldsymbol{x} + \int_{T_{11}} e^{\zeta_{11}(\boldsymbol{\lambda})} d\boldsymbol{x} \\ &= |T_{00}|\, e^{0} \quad + |T_{01}|\, e^{\lambda_5} \quad + |T_{11}|\, e^{\lambda_4 + \lambda_5} \\ &= 20 \quad + 60\, e^{\lambda_5} \quad + 120\, e^{\lambda_4 + \lambda_5} \ . \end{aligned}$$

From the optimality conditions of Eq. (6) it follows that

$$\begin{cases} \left(120\ e^{\lambda_4+\lambda_5}\right)/Z_{K_2} = & \pi_{s_4} = & 0.4 \\ \left(60\ e^{\lambda_5}+120\ e^{\lambda_4+\lambda_5}\right)/Z_{K_2} = & \pi_{s_5} = & 0.5\ , \end{cases}$$

which can be solved analytically to obtain $e^{\lambda_4^*}=2$, $e^{\lambda_5^*}=1/15$ and $Z_{K_2}=40$, giving the probability measure

$$p_{\boldsymbol{\lambda}^*}(\boldsymbol{r}) = 1/40 \cdot 2^{\chi_{s_4}(\boldsymbol{r})} \cdot 1/15^{\chi_{s_5}(\boldsymbol{r})}\ .$$

The case of certain satisfiability: A special case which allows the problem to be simplified consists of sets of statements such that all the statements are certain (i.e. $\pi_s = 1$). Indeed, we can show that the maximum entropy distribution incorporating a set of statements S all with $\pi_s = 1$ is

$$p(\boldsymbol{r}) = \begin{cases} 1/Z & \text{if } \boldsymbol{r} \text{ satisfies all } s \in S, \text{ and} \\ 0 & \text{otherwise}\ , \end{cases}$$

where $Z = |T_{\mathbf{1}}|$ is the measure of the subset $T_{\mathbf{1}}$, which consists of the regions where all the statements in S are satisfied.

B. Modeling a dataset

Our models take as input an original dataset D and a collection of logical statements S. To each deterministic row d_i in D we associate a random counterpart denoted by $\boldsymbol{r}_i$, constrained by those statements in S that are satisfied on d_i. Let us denote this subset of statements by S_i, that is $S_i = \{s \in S : \chi_s(d_i) = 1\}$. Using the row model introduced in the previous section, henceforth denoted by $p_{\texttt{ROW}}$, the probability of the row values can now be defined as

$$\mathbb{P}\left(\boldsymbol{r}_i \mid D, S\right) := p_{\texttt{ROW}}(\boldsymbol{r}_i \mid S_i)\ . \tag{12}$$

Given the row probabilities, our goal is to combine them into a probability over all possible values of a dataset.

A naïve approach would be to model the data as a set of independent rows so that the total probability is a simple product. This has a major drawback, though: the total probability depends too much on the probabilities of the individual rows. For instance, if the satisfiability probability of a statement on a row is halved, the total probability will be halved too.

Instead, our models compute the total probability as a weighted average of row models. In our first model variant, called MEALL, the average runs over all the rows in the dataset, while our second model variant, called MEBLK, considers only those rows which satisfy the given query statement.

Both MEALL and MEBLK are mixture models. MEALL is represented by the solid arrows of the graphical model in Fig. 3. Each of the M plates represents a variable $\boldsymbol{r}_i$ depending on S and d_i through S_i, as explained above. The mixing coefficient ρ acts as a prior probability over the row indices. It specifies which row models should be mixed together to obtain the final distribution over the values of $\boldsymbol{r}$, a single row summarizing the entire dataset. To enforce the selection behaviour $\boldsymbol{r} = \boldsymbol{r}_\rho$, the row selecting function can be expressed as

$$\mathbb{P}\left(\boldsymbol{r} \mid \boldsymbol{r}_1, \ldots, \boldsymbol{r}_M, \rho\right) := \delta\left(\boldsymbol{r} - \boldsymbol{r}_\rho\right), \tag{13}$$

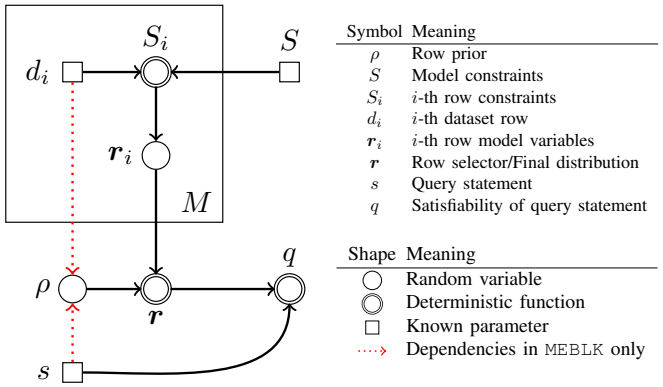


Figure 3: Graphical Representation of MEALL (solid lines) and MEBLK (all lines).

where $\delta(\cdot)$ is the Dirac delta function.[1] The row prior ρ is the uniform distribution over the row indices.

We denote by q the satisfiability of a query statement s on a row $\boldsymbol{r}$, which is constant given s and $\boldsymbol{r}$:

$$\mathbb{P}\left(q \mid \boldsymbol{r}, s\right) := \chi_s(\boldsymbol{r})\ . \tag{14}$$

Now the posterior distribution of MEALL becomes

$$\mathbb{P}\left(q \mid s; D, S\right) = \sum_{\rho=1}^{M} \mathbb{P}\left(\rho\right) \int_{\mathcal{V},\mathcal{V}_1,\ldots,\mathcal{V}_M} \mathbb{P}\left(\boldsymbol{r} \mid \boldsymbol{r}_1, \ldots, \boldsymbol{r}_M, \rho\right) \times \mathbb{P}\left(q \mid \boldsymbol{r}, s\right) \times \prod_{i=1}^{M} \mathbb{P}\left(\boldsymbol{r}_i \mid D, S\right) d\boldsymbol{r} d\boldsymbol{r}_1 \cdots d\boldsymbol{r}_M \tag{15}$$

which simplifies by substituting the specific distributions to

$$p_{\texttt{ALL}}(q \mid s; D, S) = \frac{1}{M} \sum_{i=1}^{M} p_{\texttt{ROW}}(q \mid s; S_i)\ , \tag{16}$$

with $p_{\texttt{ROW}}(q \mid s; S_i) := \mathbb{E}\left[\chi_s(\boldsymbol{r}_i) \mid S_i\right]$ being the probability that the query statement s is satisfied on row $\boldsymbol{r}_i$.

The quantity in Eq. (16) is essentially an average over all the probabilities assigned by the rows. If we change the row prior distribution to be uniform over the rows which support the query statement s exclusively,

$$\mathbb{P}\left(\rho \mid d_1, \ldots, d_M, s\right) := \frac{\chi_s(d_\rho)}{\sum_{i=1}^{M} \chi_s(d_i)}\ , \tag{17}$$

we obtain the MEBLK model, shown as the dotted dependencies in Fig. 3.

Note that MEALL possesses the useful property that $p_{\texttt{ALL}}(q|\neg s;\cdot) = 1 - p_{\texttt{ALL}}(q|s;\cdot)$, which does not generally hold under MEBLK. The latter, however, offers a more intuitive interpretation of the occurrence probability, in accordance to the framework of De Bie [6]. In particular, it is better suited to comparing queries without penalizing support size and yields qualitatively better results, as well as generally lower computational complexity.

[1] The Dirac delta, which is the continuous equivalent of the Kronecker delta, is a generalised function that assumes an infinite mass when its argument is zero, in our case effectively ensuring that only the case of $\boldsymbol{r} = \boldsymbol{r}_\rho$ is possible.

V. Algorithms

The models described in Section IV can be used for different tasks which include generating synthetic datasets and ranking patterns. We focus on the latter, which in turn involves two main operations: (i) training the model by incorporating new information in the form of patterns and (ii) querying the model, i.e. evaluating the occurence probability for a pattern. In this section, we present the algorithmic procedures for carrying out these two main operations in practice.

Recall that the patterns considered here consist of logical statements (redescriptions) and their supporting rows.

A. Training the model

As explained in the previous section, our models are mixtures of row models. Each of the row models is maintained in a factorized form where different factors involve disjoint sets of variables that do not interact in any statements, so as to allow independent computations. In addition, any given factor is shared by only a subset of the row models.

Our models are maintained as a set of factors $\mathcal{F}$. Each factor f contains a set of statements S_f and applies to a subset of rows I_f. Accordingly, it is represented as the pair $f = (S_f, I_f)$. Overloading the notation, we denote by $\text{vars}(f)$ the set of data variables associated with f, that is, $\text{vars}(f) = \bigcup_{s \in S_f} \text{vars}(s)$. For any given factor f, $\text{vars}(f)$ and I_f define a tile in the dataset, and

$$J_i = \{f \in \mathcal{F} \mid \text{vars}(f) \cap \text{vars}(s) \neq \emptyset,\ i \in I_f\} \tag{18}$$

is the set of all factors which overlap with a statement s and contain row i in their tile. Note that factor tiles do not overlap by construction. For any given statement s, $\text{vars}(s)$ and $\text{supp}(s)$ also define a tile in the dataset.

Training the model entails incrementally incorporating new statements into it. At each step, the task of the main training procedure is to update the model factors with a newly appended statement, while updating the set of factors so that independence is preserved by avoiding overlaps.

Algorithm 1: TrainModel

input : model $\mathcal{F}$, new statement s
output : updated model $\mathcal{F}$

1 $\mathcal{J} \leftarrow \{J_i : i \in \text{supp}(s)\}$; // See Eq. (18)
2 **foreach** $J \in \mathcal{J}$ **do**
3 $\quad I_J \leftarrow \bigcap_{f \in J} I_f$; // Collect cluster rows
4 $\quad \mathcal{F} \leftarrow \mathcal{F} \cup \left\{\left(\{s\} \cup \bigcup_{f \in J} S_f, I_J\right)\right\}$; // Add new factor
5 $\quad$ **foreach** $f \in J$ **do** // Update overlapping factors
6 $\quad\quad I_f \leftarrow I_f \setminus I_J$; // Update rows
7 $\quad\quad$ **if** $I_f \neq \emptyset$ **then**
8 $\quad\quad\quad \mathcal{F} \leftarrow \mathcal{F} \setminus \cup\{f\}$; // Delete
9 **return** $\mathcal{F}$;

The pseudo-code for this procedure, TrainModel, is presented in Alg. 1. It works as follows. When adding a new statement s to the model, we form the collection $\mathcal{J}$ of the sets of existing factors that overlap with s, as per Eq. (18) (line 1). Then, for each cluster J we create a new factor and add it to the model (line 4). The new factor applies exactly to the rows of the cluster (line 3) and contains all statements from the existing overlapping factors, in addition to s. We also update the set of rows to which the existing overlapping factors apply (line 6), deleting the factor altogether if it no longer applies to any row (line 8).

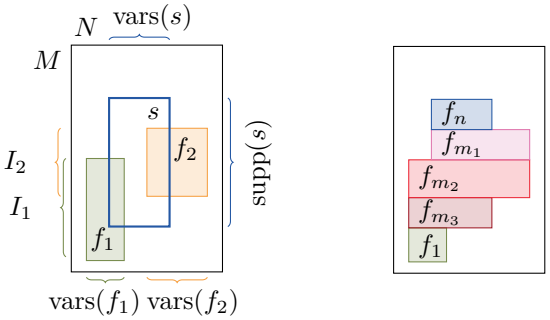


Figure 4: From two existing factors and a new statement (left) to the updated model containing five factors (right).

In the example shown in Fig. 4, the model contains two factors, f_1 and f_2. A new statement s is to be added, triggering the creation of several new factors. Factors f_{m_1}, f_{m_3} and f_{m_2} are created by merging the rows where s overlaps respectively with f_1, with f_2, and with both, while factor f_n is created for the rows that did not overlap with either (corresponding in Alg. 1 to the case of an empty row cluster $J = \emptyset$). The updated f_1, now covering fewer rows, remains, whereas f_2 has been deleted.

Note that the type of the domain of the variables only affects TrainModel implicitly, through the satisfiability assessment $\chi_s(\cdot)$ of the newly appended statement s over the dataset rows.

B. Querying the model

We now turn to the procedure that allows us to query a model. That is, given a model that has been trained as explained above, the original dataset and a pattern extracted from the dataset, the aim is to evaluate the occurence probability of the pattern.

The main building block in computing this probability is the evaluation of the occurrence probabilities over single rows, essentially the probabilities predicted by the row models.

Algorithm 2: QueryModelRow

input : model $\mathcal{F}$, statement s
output : probability $p_{\text{ROW}}(s|D;\mathcal{F})$

1 $\mathcal{F}_s \leftarrow \{f \in \mathcal{F} | \text{vars}(f) \cap \text{vars}(s) \neq \emptyset\}$;
2 $(\boldsymbol{w}, \boldsymbol{v}) \leftarrow$ Quantization$(s, \mathcal{F}_s)$;
3 $Z_s \leftarrow \mathbf{1}^T \boldsymbol{w}$;
4 $\boldsymbol{p} \leftarrow (\boldsymbol{v} \otimes \boldsymbol{w})/Z_s$; // $\otimes$ is the element-wise product
5 $p \leftarrow$ MarginalizeAndNormalize$(s, \mathcal{F}_s, \boldsymbol{p})$;
6 **return** p;

Querying a row model: Each row model is parameterised by the subsets of statements that occur on the corresponding data row. The occurrence probability $p_{\text{ROW}}(s \mid D; \mathcal{F})$ of a statement for a given row model is computed by QueryModelRow as shown in Alg. 2 and works as follows.

At first, we collect in $\mathcal{F}_s$ the factors of the model that share variables with statement s. Next, the domain of the variables can be partitioned into regions over which the value of the characteristic function of s is constant, by syntactically parsing the statement and collecting all the involved thresholds (cf. Sec. IV-A). Since, by construction, the value of the characteristic function of s is the same in all points of a region, we can compactly represent the result of this quantisation by a pair of vectors $(\boldsymbol{w}, \boldsymbol{v})$, where $\boldsymbol{w}_i$ measures the area of region i and $\boldsymbol{v}_i$ indicates whether s is satisfied on that region (line 2). Using this information, we are able to compute the probabilities for the different regions satisfying the statement (lines 3–4). Finally, the probability of each region of the domain is appropriately re-weighted with respect to the different factors in $\mathcal{F}_s$ (line 5). More specifically, statement s together with the subset $\mathcal{F}_s$ form a Junction Tree [3], and we may therefore employ the Message Passing steps of the Junction tree algorithm to efficiently re-scale the initial probabilities. In this way, the effect of each overlapping factor is taken into account and the corresponding constraints respected, to finally yield the probability of statement s.

Putting the rows together: Now that we know how to compute the probabilities for individual rows, we need to combine them together. The two models, `MEBLK` and `MEALL`, offer two alternatives for doing so. The overall probability returned by `MEALL` is simply the average of $p_{\texttt{ROW}}(s \mid D; \mathcal{F}_i)$ over all rows i in the dataset, while `MEBLK` only averages over rows that satisfy the statement under evaluation, i.e. over $\mathrm{supp}(s)$.

C. A ranking scheme

Combining the two main operations of training and evaluation explained above, the procedure for ranking patterns is presented in Alg. 3.

Algorithm 3: RANKPATTERNS

```
   input  : dataset D, set of statements S with selected statement s_0
   output : ordered list of statements O
1  F ← TRAINMODEL(∅, s_0);
2  O ← (s_0); S ← S \ {s_0};
3  while S ≠ ∅ do
4  |   s* ← s ∈ S, minimizing QUERYMODEL(F, D, s);
5  |   F ← TRAINMODEL(F, s*);
6  |   O ← (O, s*); S ← S \ {s*};
7  return O;
```

This procedure takes as input a dataset and a collection of statements, with one of them designated to initialize the model training (line 1). This method then iteratively constructs a ranking of all the statements. In each step, the model is queried to identify the statement with lowest predicted probability (line 4), which is essentially the statement whose observation is most surprising at that point. This statement is incorporated into the model (line 5), appended to the list of results and removed from the set of candidates (line 6). The procedure iterates until all statements have been ranked. Note that this method closely resembles the dynamic iterative ranking scheme of Mampaey et al. [23].

VI. EXPERIMENTAL EVALUATION

In this section, we present experiments to investigate the behaviour and performance of our algorithms and compare our two models `MEALL` and `MEBLK`.

We implemented our algorithm[2] using MATLAB for the high level procedures and C++ for the core operations. All experiments were run on a cluster with 16 cores (at 2.4 GHz and with 48 GB of memory). The sets of redescriptions for the real-world data were mined in advance using the REREMI algorithm [10] in the Siren redescription mining interface[3] [11], [12]. Among existing redescription mining algorithms, REREMI offers the most flexible query language and is simultaneously rather suceptible to redundancy in the result set.

We proceed with a series of experiments on synthetic datasets, before moving on to real-world data.

A. Evaluation on synthetic datasets

Our goal in this series of experiments with synthetic datasets is to shed light on the different aspects that impact the complexity of the computations in a controlled experimental setting. Our focus here is on the quantitative evaluation of the performance of the algorithms and our primary measure in these experiments is therefore the wall-clock time for training and querying the model.

We start by considering the simplest case of a model trained with a single Boolean statement and queried with another Boolean statement. That is, we consider two statements, s_t and s_q, used to constrain and to query the model, respectively. In other words, we first train the model with s_t, then query the trained model to evaluate the occurrence probability of s_q.

We call *width* of a statement the number of distinct variables it contains, i.e. $\mathrm{width}(s_t) = |\mathrm{vars}(s_t)|$, while the *overlap* between two statements is simply the number of variables they have in common, i.e. $\mathrm{over}(s_t, s_q) = |\mathrm{vars}(s_t) \cap \mathrm{vars}(s_q)|$. We can fully control these parameters by using logical conjunctions and choosing suitable sets of variables for our statements.

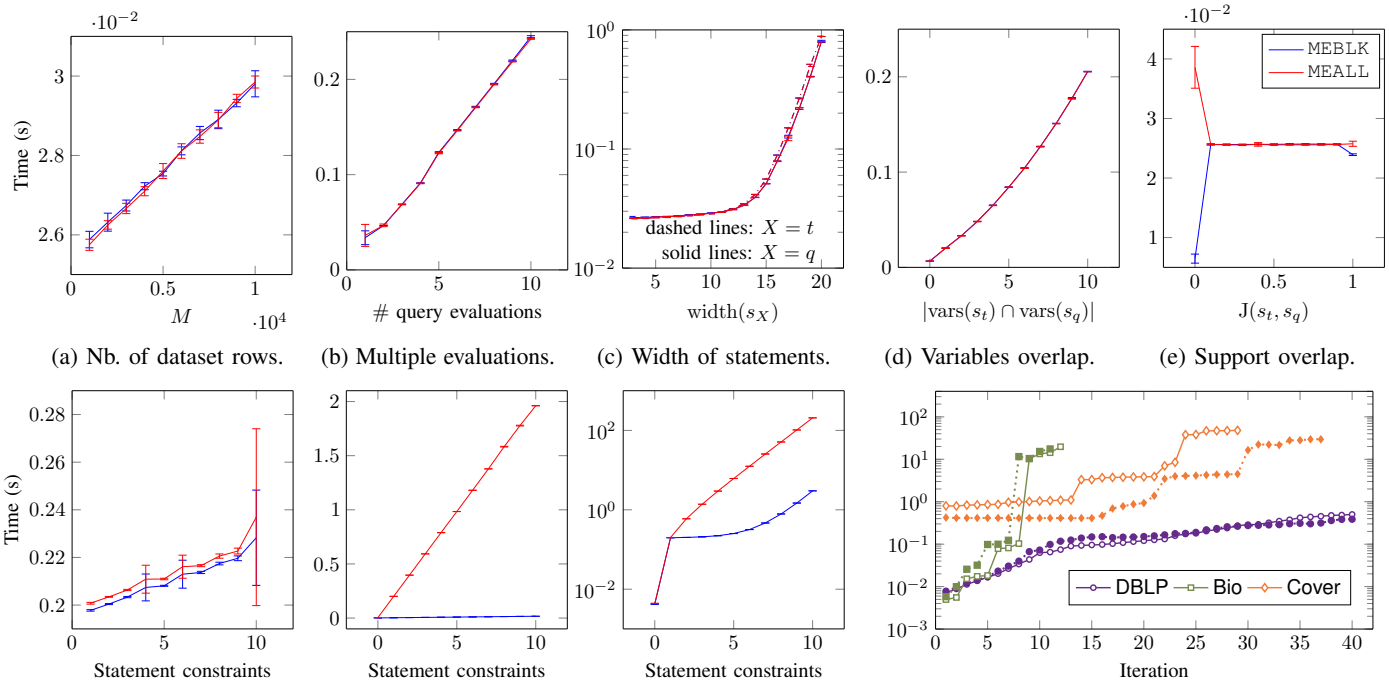
Our base case is as follows. We let $s_t = v_A \wedge v_B \wedge v_C$ and $s_q = v_B \wedge v_C \wedge v_D$, so that $\mathrm{width}(s_t) = \mathrm{width}(s_q) = 3$ and $\mathrm{over}(s_t, s_q) = 2$. We fix the number of rows in our dataset to $M = 1000$, of which 70% satisfy both s_t and s_q while the rest are evenly distributed over the remaining 3 truth combinations.

Starting from this base case, we can study the impact of different parameters in turn, by repeating the training and querying of our model while varying a chosen parameter of the problem. For each configuration, we record the total running times for training and querying. The results are reported in Fig. 5. The markers and the error bars indicate respectively the mean running times and twice the standard deviation for 10 repetitions of each configuration.

The first parameter we consider is the number of rows in the dataset (Fig. 5a), the second parameter is the number of query evaluations, that is, we train the model once with s_t, before

[2]The source code is available at http://siren.mpi-inf.mpg.de/max-ent/.

[3]http://siren.gforge.inria.fr



(a) Nb. of dataset rows. (b) Multiple evaluations. (c) Width of statements. (d) Variables overlap. (e) Support overlap.

(f) Nb. of row clusters: constant. (g) Nb. of row clusters: linear growth. (h) Nb. of row clusters: exponential growth. (i) Real world results: mean runtime per query for MEALL (solid lines) and MEBLK (dashed lines).

Figure 5: Experimental results: (a-h) synthetic datasets (running times for training and querying averaged over 10 repetitions, the width of the error bars equals twice the standard deviation) and (i) real world datasets (averaged querying times).

repeatedly querying it with the same statement s_q (Fig. 5b). In both cases, the behaviour of both models is linear.

Next, we study the impact of the shape of the statements by fixing the width of one statement to 3, while increasing that of the other, always keeping their overlap constant. The models show the same exponential increase of time in both cases (Fig. 5c). Indeed, in both cases factors of equal sizes need to be evaluated.

We look at the impact of interactions between the statements, in terms of shared variables and of shared rows. The impact of varying variable overlap, as measured by $\mathrm{over}(s_t, s_q)$, with constant widths $\mathrm{width}(s_t) = \mathrm{width}(s_q) = 11$, is shown in Fig. 5d. On the other hand, the impact of varying row overlap, as measured by $\mathrm{J}(s_t, s_q)$, is shown in Fig. 5e. The latter shows further evidence that the number of different value combinations matters, rather than the number of rows in the overlap. Indeed, since the algorithm processes rows clustered according to the combination of factors they participate in, the number of such clusters in the dataset has a major impact on the running times.

To further investigate the complexity arising from the structure of the dataset through the number of clusters it induces, we modify our setup as follows. Starting with an empty model, in each iteration i we add a different statement s_i then query the updated model with the same statement s_q. All statements are defined over 10 Boolean variables. The query statement is a simple conjunction $s_q = v_1 \wedge \ldots \wedge v_{10}$, whereas the training statement added at the i-th iteration has the form $s_i = v_i \wedge \left(\left(\bigvee_{\kappa \neq i} v_\kappa\right) \vee \neg \left(\bigvee_{\kappa \neq i} v_\kappa\right)\right)$. The training statements s_i all have $\mathrm{width}(s_i) = 10$ and $\mathrm{over}(s_i, s_q) = 10$, but have all distinct truth tables.

Next, we create 3 different datasets, each with $M = 1024$ rows. The first one only contains rows from $\{\mathbf{0}, \mathbf{1}\}$, i.e. with all of their values equal, so that at most 2 clusters will be created, regardless of the number of training statements added to the model. The second dataset contains rows from the standard basis $\{\boldsymbol{e}_\kappa : \kappa = 1, \ldots, 10\}$, so that the number of clusters increases by one with each added training statement. Finally, the third dataset contains rows with all the possible Boolean combinations of the 10 variables, so that the number of clusters grows exponentially, reaching 2^i at the i-th iteration and thus representing the worst case scenario.

The running times for each iteration on these three datasets are reported in Fig. 5f-5h, respectively. We observe that the running time for the MEALL model closely follows the number of clusters, since they all have to be evaluated. On the other hand, the MEBLK model only evaluates the clusters that support s_q. The small additional overhead for the MEBLK model in the last case is due to the training phase, which still needs to track all clusters.

B. *Evaluation on real-world datasets*

After this systematic investigation of the performance of our algorithms and models on synthetic datasets, we now present results from real-world data for a qualitative assessment.

Our first example illustrates the iterative ranking process. Using the ReReMi algorithm, we mined redescriptions from

Table III: Ranking DBLP with MEBLK. Steps 0, 1, 6 and 7.

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	J(R)
SDM	P. Yu ∨ V. Kumar	.198
SDM	P. Yu ∨ V. Kumar	.198
COLT ∧ ICML	R. Schapire	.175
COLT ∧ ICML	R. Schapire	.175
COLT	P. Bartlett ∨ M. Kearns	.173
COLT	P. Bartlett ∨ A. Blum	.172
SDM	J. Han ∨ V. Kumar	.166
ICDT	R. Miller∨S. Abiteboul	.164
SDM	V. Kumar ∨ H. Wang	.164
COLT	C. Smith ∨ Y. Mansour	.162

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	$p(R)$
SDM	P.S.Yu∨V. Kumar	—
EDBT ∧ PODS	A. Silberschatz	.125
FOCS ∧ SODA	N. Alon	.125
UAI ∧ ICML	D. Koller	.125
COLT ∧ ICML	R. Schapire	.125
COLT ∧ ICML	M. Kearns	.125
COLT ∧ ICML	R. Schapire	.125
ICDT ∧ PODS	S. Abiteboul	.125
VLDB ∧ ICDT	H. Garcia-Molina	.125
COLT ∧ ICML	A. Blum	.125

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	$p(R)$
SDM	P. Yu ∨ V. Kumar	—
EDBT ∧ PODS	A. Silberschatz	—
FOCS ∧ SODA	N. Alon	—
UAI ∧ ICML	D. Koller	—
VLDB ∧ ICDT	H. Garcia-Molina	—
COLT ∧ ICML	M. Kearns	.133
COLT ∧ ICML	A. Blum	.134
FOCS ∧ COLT	Y. Mansour	.143
COLT ∧ ICML	R. Schapire	.147
COLT ∧ ICML	R. Schapire	.147

$q_{\mathbf{L}}$	$q_{\mathbf{R}}$	$p(R)$
SDM	P. Yu ∨ V. Kumar	—
EDBT ∧ PODS	A. Silberschatz	—
FOCS ∧ SODA	N. Alon	—
UAI ∧ ICML	D. Koller	—
VLDB ∧ ICDT	H. Garcia-Molina	—
COLT ∧ ICML	M. Kearns	—
ICDT∧SIGMOD	S. Abiteboul	.153
ICDT ∧ PODS	S. Abiteboul	.153
FOCS ∧ COLT	Y. Mansour	.161
SDM ∧ KDD	J. Han	.215

Table IV: Redescriptions from Bio ranked by accuracy.

	$q_{\mathbf{L}}$	$q_{\mathbf{R}}$
(1)	Polar Bear	$[-7.07 \leq t_5 \leq -3.38]$
(2)	Polar Bear	$[-16.7 \leq t_3 \leq -11.5]$
(3)	Polar Bear	$[-4.5 \leq t^+_{10} \leq -1]$
(4)	Polar Bear	$[1 \leq t^+_9 \leq 3.5]$
(5)	Polar Bear	$[-9.6 \leq t^+_4 \leq -5.6]$
(6)	Polar Bear	$[-11.9 \leq t^+_3 \leq -7.3]$
(7)	B. Vole ∨ N.R. Vole ∨ S. Mouse ∨ H. Seal	$[10.9 \leq t^+_8 \leq 29.9] \wedge [-9.2 \leq t^+_{12} \leq 12.8] \wedge [34.7 \leq p_6] \wedge [47.6 \leq p_8]$
(8)	W. Mouse	$(([2.9 \leq t^+_3] \vee [9.7 \leq t^+_7 \leq 13.2]) \wedge [-3.26 \leq t_{11} \leq 15.9]) \vee [5.81 \leq t_6 \leq 5.88]$
(9)	W. Mouse ∨ H. Seal ∨ A. Noctule	$[-0.8 \leq t^+_2] \wedge [-0.141 \leq t_{10} \leq 19.6] \wedge [26.6 \leq p_4]$

Table V: Redescriptions from Bio ranked with MEALL.

	$q_{\mathbf{L}}$	$q_{\mathbf{R}}$
(1)	Polar Bear	$[-7.07 \leq t_5 \leq -3.38]$
(2)	G.W. Shrew ∧ E. Mongoose	$([15.6 \leq t^-_8 \leq 19] \wedge [1.62 \leq p_8 \leq 7.44] \wedge [66.2 \leq p_{12} \leq 137]) \vee [13.9 \leq t_3 \leq 14.3]$
(3)	W. Mouse ∧ N. Bat ∧ E.P. Shrew	$([3.2 \leq t^+_3 \leq 14.5] \wedge [17.3 \leq t^+_8 \leq 25.2] \wedge [14.9 \leq t^+_9 \leq 22.8]) \vee [19.6 \leq t_7 \leq 19.9]$
(4)	Wolverine	$([7.2 \leq t^+_9 \leq 11.7] \wedge [-11. \leq t_3 \leq -5.37] \wedge [63.1 \leq p_7 \leq 106]) \vee [-3.43 \leq t_{11} \leq -3.34]$
(5)	H. Mouse ∧ E. Mole	$[-0.3 \leq t^-_4 \leq 8.7] \wedge [19.4 \leq t^+_8 \leq 27.2] \wedge [45.4 \leq p_6] \wedge [48.8 \leq p_8 \leq 126]$
(6)	W. Lemming	$(([-4.3 \leq t^+_1 \leq 1.6] \wedge [3.29 \leq t_5 \leq 9.75]) \vee [-6.8 \leq t^-_3 \leq -6.8]) \wedge [21.9 \leq p_3 \leq 72.2]$
(7)	N. lemming	$(([-21.8 \leq t^-_2 \leq -8.7] \wedge [12.5 \leq t^+_8 \leq 16.6]) \vee [5.41 \leq t_5 \leq 5.43]) \wedge [59. \leq p_8 \leq 166]$
(8)	E.P. Vole ∨ R. Muntjac	$[-0.8 \leq t^+_2 \leq 9] \wedge [10.9 \leq t^+_4 \leq 17.5] \wedge [17.7 \leq t^+_9 \leq 24.4] \wedge [43.3 \leq p_7]$
(9)	L. Shrew	$(([-9.7 \leq t^+_2 \leq -4.7] \vee [4.44 \leq t_{10} \leq 4.52]) \wedge [41.7 \leq p_6 \leq 68.3]) \vee [-4.39 \leq t_{12} \leq -4.32]$

the DBLP dataset. This dataset is extracted from the popular computer science bibliography.[4] The entities are researchers and one side records major conferences where they published, while the other side records the co-authorship graph ($|E| = 2345$, $|V_{\mathbf{L}}| = 19$, and $|V_{\mathbf{R}}| = 2345$; both sides are Boolean). In Tab. III we show the evolution of the top-ranked redescriptions over a few steps.

Initially, the redescriptions fed to the algorithm are sorted by accuracy. In the first step, the top redescription is added to the model, the probabilities for the other redescriptions are computed and the list is sorted by increasing occurrence probability, since higher probabilities are associated to less surprising and thus less interesting redescriptions. The second redescription is then added to the model, the probabilities recomputed and the ranking updated. The iterations continue until the entire list has been processed.

Tab. III shows how the redundant redescriptions are pushed away from the top of the list. For example, after adding the first redescription, all other redescriptions with SDM on the left-hand side are pushed out of the table. Notice, however, that *some* overlap is possible, if the redescriptions are otherwise surprising enough: for example, in the 7th step, ICML appears in the right-hand side of two redescriptions that have already been included in the model.

Our second dataset, Bio, comes from the domain of ecology. The entities represent geographic areas of Europe, the left-hand side records the presence of various mammals species [26], while the right-hand side consists of bioclimatic variables, that is, monthly average rainfall and monthly average, minimum, and maximum temperatures [15] ($|E| = 2575$, $|V_{\mathbf{L}}| = 194$ and $|V_{\mathbf{R}}| = 48$; the species records are Boolean and the climate variables real-valued).

In Tab. IV we show the top of the list of redescriptions, sorted by accuracy, which were fed to the ranking procedure. The top of the output list is shown in Tab. V. The top redescriptions selected by our algorithm exhibit a much greater diversity. In particular, the redescriptions about the Polar Bear do not take all the top spots any longer: albeit very accurate, they are highly redundant and of limited interest when taken together.

Our last dataset, Cover[5], also comes from the domain of ecology. The entities represent geographic areas of a national forest in Colorado, USA. The wilderness area, soil type, and cover type constitute the right-hand side variables, while other, topographic variables, such as elevation and slope, are on the left-hand side. ($|E| = 581012$, $|V_{\mathbf{L}}| = 10$ and $|V_{\mathbf{R}}| = 45$; all variables on the right are Boolean except for the cover type which is nominal, and all those on the left are real-valued).

The RANKPATTERNS algorithm was run to rank sets of redescriptions of size $|S| = 100$, 60 and 230, extracted from DBLP, Bio and Cover respectively. For each dataset, Fig. 5i depicts the average time required to query the occurence probability for each of the remaining candidate redescriptions, during each iteration of the ranking algorithm, ignoring the negligible time required for training the model.

These plots show steep rises in the query times that correspond to iterations where the last training redescription

[4]http://www.informatik.uni-trier.de/~ley/db/

[5]https://archive.ics.uci.edu/ml/datasets/Covertype

overlaps with the largest factor in the model, which must therefore be extended. As explained in Sec. IV, our algorithms use factorisation and quantisation to reduce the computational cost of evaluating occurence probabilities. For this reason, the runtime complexity is dominated by the size of the largest factor in terms of value combinations, which may grow arbitrarily large as the degree of overlap increases. This effect is exacerbated in the case of numerical variables due to the quantisation becoming finer following some model updates.

Fortunately, statements involving variables that have occurred in earlier selected statements are typically assigned higher probabilities. This means that such overlapping redescriptions are generally deemed uninteresting and pushed lower in the ranking, with the beneficial consequence of delaying the formation of larger factors to later iterations of the algorithm.

VII. Conclusions

Thus far, redescription mining was mostly focused on the problem of finding good redescriptions, ignoring the more global problem of finding a good *set* of redescriptions. In this work, we have approached this latter problem from the point of view of maximum-entropy distributions: a redescription is non-redundant if and only if it has a low likelihood under the maximum-entropy distribution, conditioned on the redescriptions previously seen. Thus, our approach fits into the general framework of De Bie [6].

However, working with redescriptions comes with its own challenges. Most notably, restricting redescriptions only to conjunctive queries – as is (implicitly) done in the existing work on subjective interestingness – severely limits the usability of the method. Therefore, we had to develop an approach that can also handle disjunctive queries.

Another significant difference with the existing line of work on subjective interestingness is that we do not enforce any a priori conditions on the data (such as row and column marginals). Incorporating such constraints into our model is an interesting topic for future work.

Finally, another natural direction for future work is to develop methods that can directly mine the most surprising redescription given the current model; the work presented in this paper can only be applied as a post-processing step.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. *SIGMOD Record*, 27(2):94–105, 1998.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB'94*, pages 487–499, 1994.

[3] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[4] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A Maximum Entropy Approach to Natural Language Processing. *Comput. Linguist.*, 22(1):39–71, 1996.

[5] S. Bickel and T. Scheffer. Multi-View Clustering. In *ICDM'04*, pages 19–26, 2004.

[6] T. De Bie. Maximum Entropy Models and Subjective Interestingness: An Application to Tiles in Binary Databases. *Data Min. Knowl. Disc.*, 23(3):407–446, 2011.

[7] T. Edwin. Jaynes. Probability Theory: The Logic of Science. *Cambridge university press*, 10:33, 2003.

[8] E. Galbrun. *Methods for Redescription Mining*. PhD thesis, University of Helsinki, Helsinki, 2013.

[9] E. Galbrun and A. Kimmig. Finding Relational Redescriptions. *Mach. Learn.*, 96(3):225–248, 2014.

[10] E. Galbrun and P. Miettinen. From Black and White to Full Color: Extending Redescription Mining Outside the Boolean World. *Stat. Anal. Data Min.*, 5(4):284–303, 2012.

[11] E. Galbrun and P. Miettinen. Siren: An interactive tool for mining and visualizing geospatial redescriptions. In *KDD*, pages 1544–1547, 2012.

[12] E. Galbrun and P. Miettinen. Interactive Redescription Mining. In *SIGMOD'14*, pages 1079–1082, 2014.

[13] A. Gallo, P. Miettinen, and H. Mannila. Finding Subgroups having Several Descriptions: Algorithms for Redescription Mining. In *SDM'08*, pages 334–345, 2008.

[14] A. J. Grove, J. Y. Halpern, and D. Koller. Random worlds and maximum entropy. In *LICS'92*, pages 22–33, 1992.

[15] R. J. Hijmans, S. E. Cameron, L. J. Parra, P. G. Jones, and A. Jarvis. Very High Resolution Interpolated Climate Surfaces for Global Land Areas. *International Journal of Climatology*, 25:1965–1978, 2005.

[16] S. Jaroszewicz and D. A. Simovici. Pruning Redundant Association Rules using Maximum Entropy Principle. In *PAKDD'02*, pages 135–147, 2002.

[17] F. V. Jensen and F. Jensen. Optimal Junction Trees. In *UAI'94*, pages 360–366, 1994.

[18] K.-N. Kontonasios and T. De Bie. Formalizing Complex Prior Information to Quantify Subjective Interestingness of Frequent Pattern Sets. In *IDA'12*, pages 161–171, 2012.

[19] K.-N. Kontonasios and T. De Bie. Subjectively interesting alternative clusterings. *Mach. Learn.*, 98(1-2):31–56, 2015.

[20] K.-N. Kontonasios, J. Vreeken, and T. De Bie. Maximum Entropy Modelling for Assessing Results on Real-Valued Data. In *ICDM'11*, pages 350–359, 2011.

[21] K.-N. Kontonasios, J. Vreeken, and T. De Bie. Maximum Entropy Models for Iteratively Identifying Subjectively Interesting Structure in Real-Valued Data. In *ECML/PKDD'13*, pages 256–271, 2013.

[22] P. Kröger and A. Zimek. Subspace Clustering Techniques. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 2873–2875. Springer, 2009.

[23] M. Mampaey, N. Tatti, and J. Vreeken. Tell Me What I Need To Know: Succinctly Summarizing Data with Itemsets. In *KDD'11*, pages 573–581, 2011.

[24] M. Mampaey, J. Vreeken, and N. Tatti. Summarizing Data Succinctly with the Most Informative Itemsets. *ACM Trans. Knowl. Disc. Data*, 6(4):16:1–16:42, 2012.

[25] H. Mannila, D. Pavlov, and P. Smyth. Prediction with Local Patterns Using Cross-entropy. In *KDD'99*, pages 357–361, 1999.

[26] A. J. Mitchell-Jones et al. *The Atlas of European Mammals*. Academic Press, 1999.

[27] P. K. Novak, N. Lavrač, and G. I. Webb. Supervised Descriptive Rule Discovery: A Unifying Survey of Contrast Set, Emerging Pattern and Subgroup Mining. *J. Mach. Learn. Res.*, 10:377–403, 2009.

[28] L. Parida and N. Ramakrishnan. Redescription Mining: Structure Theory and Algorithms. In *AAAI'05*, pages 837–844, 2005.

[29] D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. *IEEE Trans. Knowl. and Data Eng.*, 15(6):1409–1421, 2003.

[30] S. J. Phillips, R. P. Anderson, and R. E. Schapire. Maximum Entropy Modeling of Species Geographic Distributions. *Ecol. Model.*, 190(3):231–259, 2006.

[31] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, and R. F. Helm. Turning CARTwheels: an Alternating Algorithm for Mining Redescriptions. In *KDD'04*, pages 266–275, 2004.

[32] N. Tatti. Maximum entropy based significance of itemsets. *Knowl. Inf. Sys.*, 17(1):57–77, 2008.

[33] M. van Leeuwen and E. Galbrun. Association discovery in two-view data. *IEEE Trans. Knowl. and Data Eng.*, 27(12):3190–3202, 2015.

[34] J. Vreeken and M. van Leeuwen. KRIMP: Mining itemsets that compress. *Data Min. Knowl. Disc.*, 23:169–214, 2011.

[35] C. Wang and S. Parthasarathy. Summarizing Itemset Patterns Using Probabilistic Models. In *KDD'06*, pages 730–735, 2006.

[36] M. J. Zaki and N. Ramakrishnan. Reasoning About Sets Using Redescription Mining. In *KDD'05*, pages 364–373, 2005.

[37] T. Zinchenko, E. Galbrun, and P. Miettinen. Mining Predictive Redescriptions with Trees [Demo]. In *ICDMW'15*, 2015.